Работа с СУБД MySQL в IntelliJ IDEA в редакции Community

Валерий Фетисов
Нежинский государственный университет им. Гоголя.
fetisval@gmail.com

Аннотация.

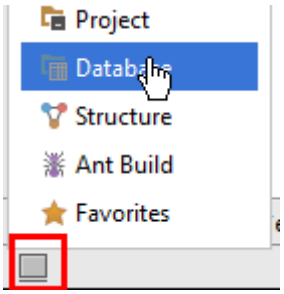
Основою статьи является лекция, которую я читаю в университете в курсе «Платформы корпоративных информационных систем». В ней рассмотрены очень слабо освещенные в печати вопросы работы с популярной СУБД MySQL в интегрированной среде разработки программного обеспечения на Java от компании JetBrains.

Ключевые слова: база данных, СУБД MySQL, MySQL, IntelliJ IDEA, среда разработки программного обеспечения на Java, Java, JDBC-драйвер, Java Data Base Connectivity, Community Edition, DataBase Navigator.

При наличии Java-приложений, получающих данные из базы данных, используется JDBC-драйвер (Java Data Base Connectivity). Именно с его помощью приложение может взаимодействовать с базой данных, хотя и не напрямую. Для каждой конкретной базы данных используется свой JDBC.  При этом java-разработчик не связан конкретной СУБД, поскольку набор методов драйвер-менеджеров для всех база данных является одинаковым.

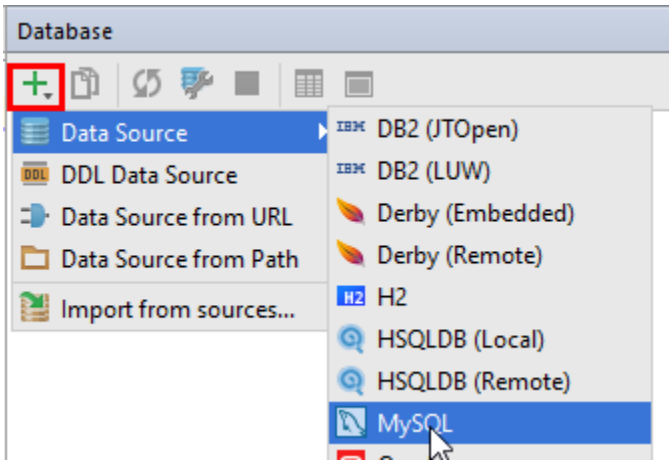
Устанавливаем соединение

Интегрировать MySQL в IIDEA довольно просто для любой ее редакции. А вот определить базу данных в качестве источника можно только в редакции Ultimate Edition. Для этого наводим курсор на небольшой прямоугольник (он включает дополнительные вкладки) в правом нижнем углу основного окна программы, что приведет к отображению перечня дополнительных вкладок программы, одним из пунктов которого является «Database».

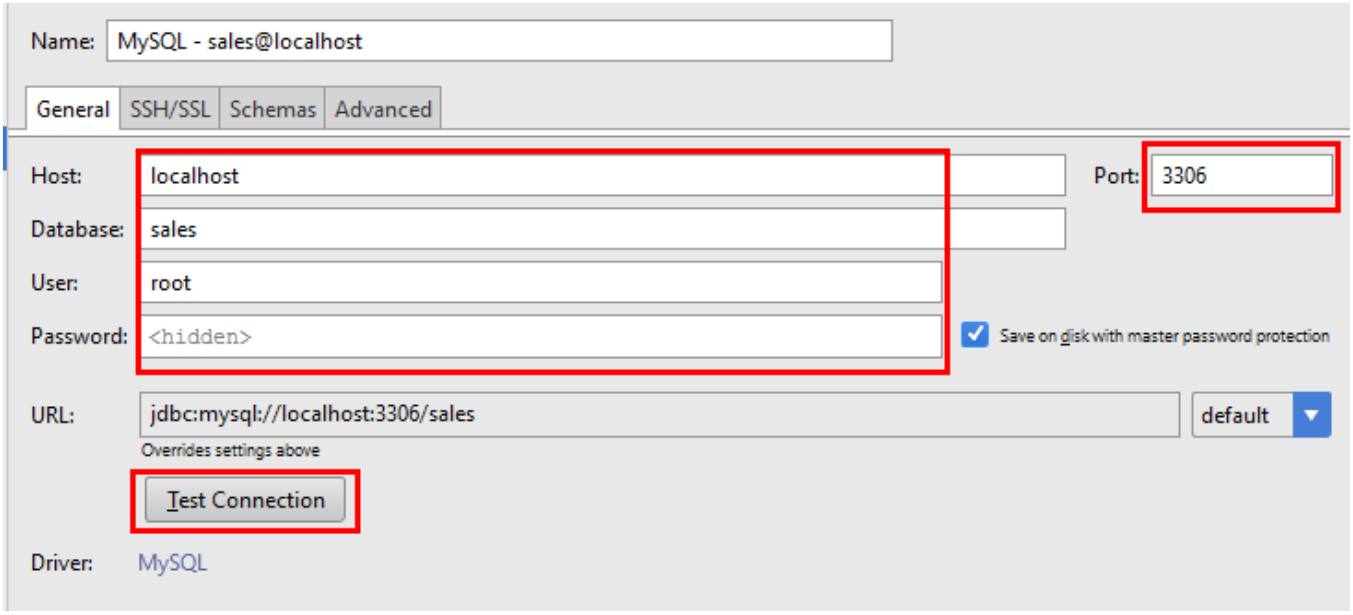


Выбор этого пункт приводит к появлению в основном окне вкладки для работы с базами данных.

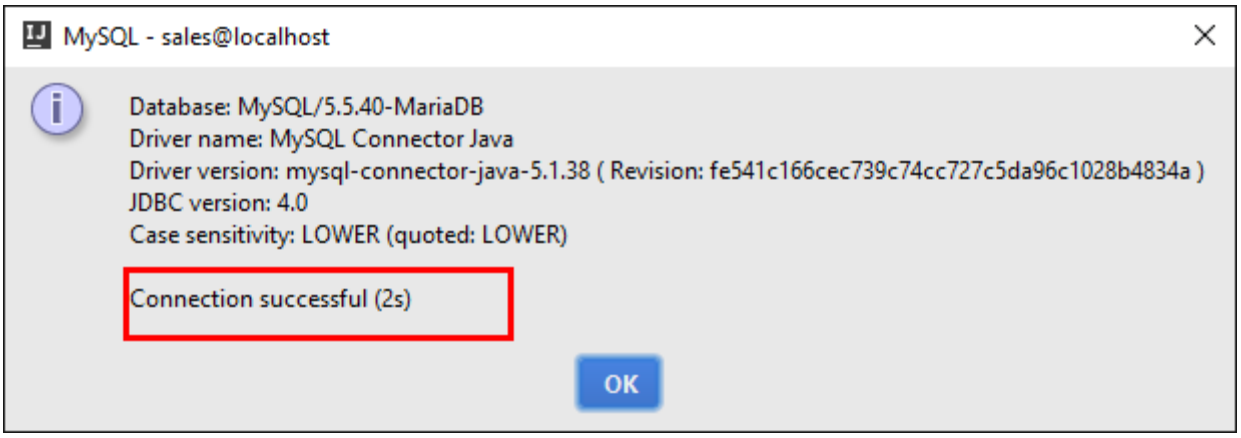
На панели инструментов вкладки «Database» раскрываем список кнопки «+» для определения источника базы данных, в качестве которого выбираем MySQL.



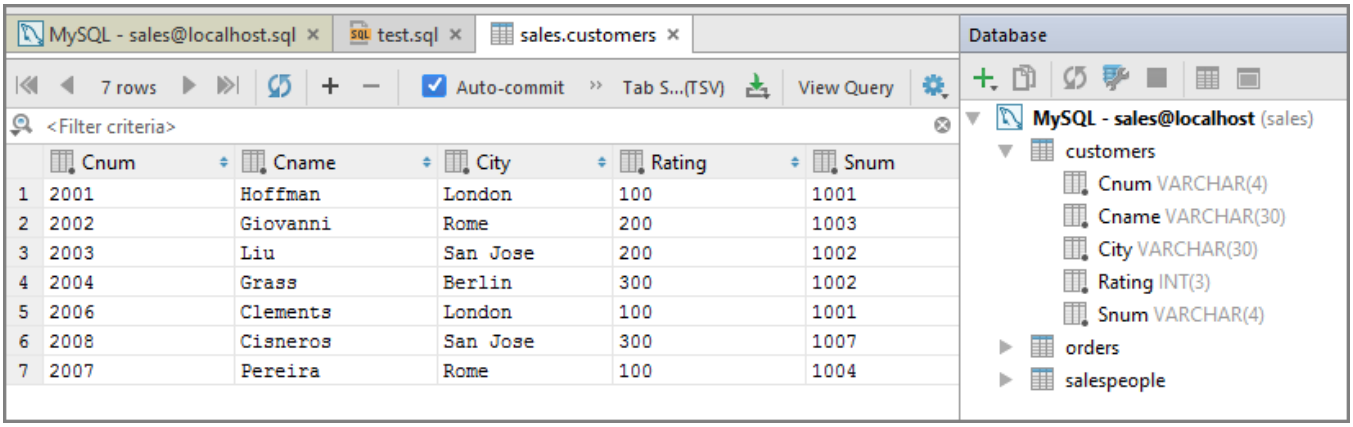
Появится окно с настройками соединения, в котором будет необходимо задать имя базы данных, пользователя и пароль. Все эти данные (а также Host и Port) соответствуют тем данным, которые использовались при установке на компьютере сервера MySQL. Кроме того, необходимо установить драйвер для работы с MySQL, который можно установить прямо из окна, щелкнув на гиперссылку Driver. Вообще же архив драйвера (mysql-connector-java-5.1.*xx*-bin.jar) можно скачать с официального сайта MySQL по ссылке http://dev.mysql.com/downloads/connector/j/.



Для того, чтобы удостовериться, что настройки соединения выполнены правильно, щелкаем кнопку «**Test Connection**», в результате чего должны получить такое:

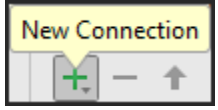


После этого во вкладке отображается выбранная база данных, с которой мы работаем практически также как и в клиенте СУБД. Вообще IIDEA (напомню: в версии Ultimate Edition) предоставляет впечатляющие средства работы с данными. Помимо модификации данных таблиц, можно модифицировать структуру самих таблиц, создавать новые, строить связи между таблицами и т.д. По большому счету наличие таких средств позволяет в простейшем случае вообще не использовать клиента СУБД.



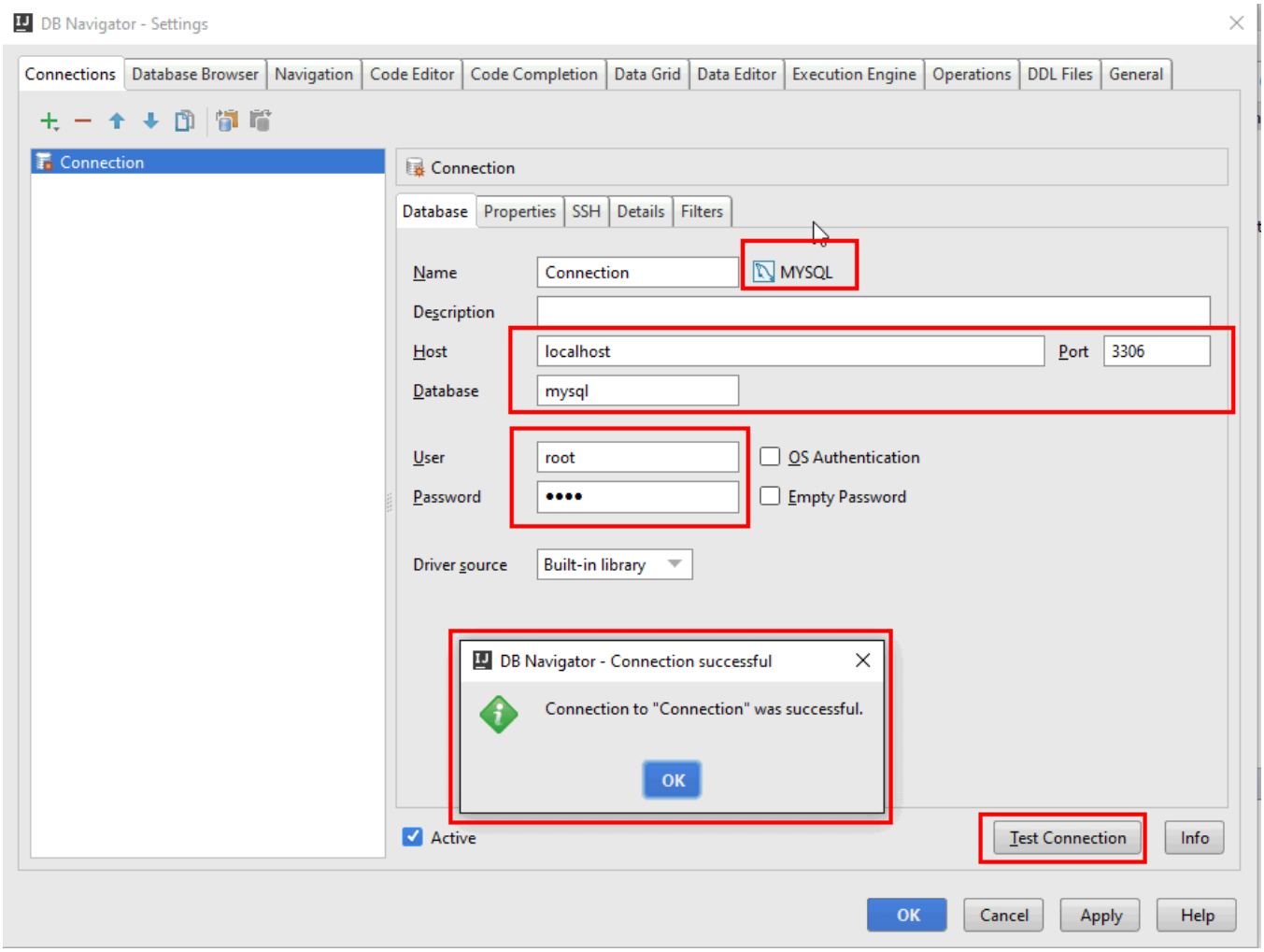
Вместе с тем, и редакция программы Community Edition позволяет достаточно комфортно работать с базой данных, в частности с MySQL. Для этого требуется просто подключить plug-in ***DataBase Navigator***. После его установки в главном меню IIDEA появится пункт DB Navigator, а на левой вкладке появится еще одна закладка «DB Browser».

Опять же первым шагом после установки ***DataBase Navigator*** следует установить соединение с базой данных. Для этого следует создать новое соединение, что может быть выполнено разными способами. Например:

1. Выполнить команду **DB Navigator ▸ Setting**, после чего откроется окно «DB Navigator Setting».
2. В этом окне открыть вкладку «Connections» (Соединения).
3. Раскрыть список кнопки 

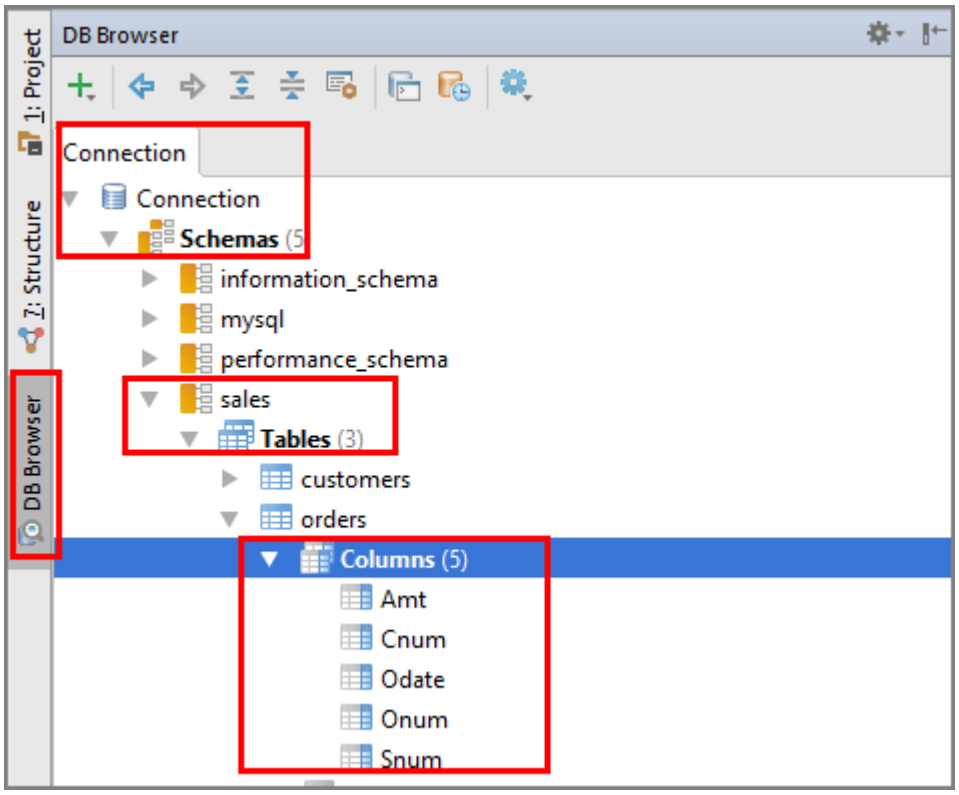
«New Connection» (Новое соединение) и добавить новое соединение для MySQL.
4. Для соединения опять-таки необходимо будет необходимо указать имя базы данных, пользователя и пароль, соответствующие соответствуют тем данным, которые использовались при установке на компьютере сервера MySQL.
5. Щелкнуть кнопку для проверки созданного соединения. Если настройки соединения

выполнены корректно, то появится сообщение «Connection to <имя_соединения> was successful».

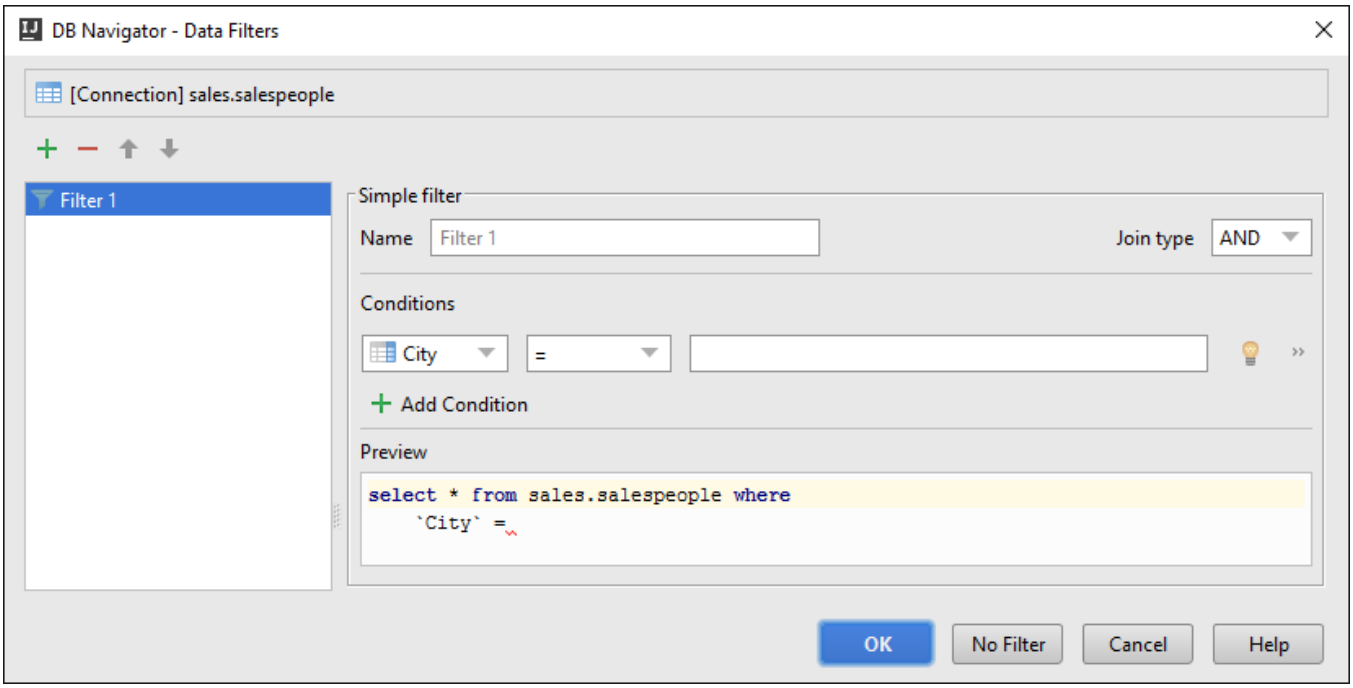


Для непосредственной визуализации данных базы данных и работы с ними следует выполнить следующее.

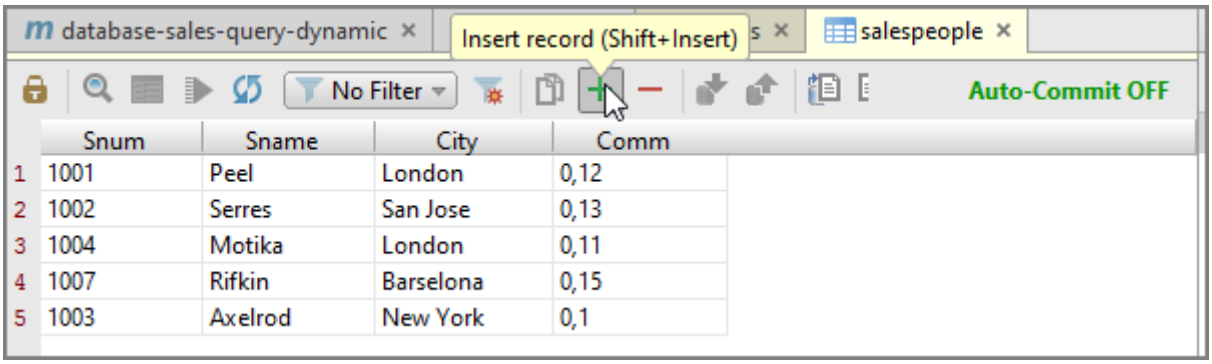
1. На левой вкладке окна открыть вкладку «DB Browser».
2. На вкладке отобразится иерархическая структура доступных в соединении база данных и их содержимого вплоть до полей. Попутно. Для определения типа данных и разрядности поля необходимо просто навести на него курсор.



3. Вызвать контекстное меню на названии таблицы, данные которой нужно визуализировать и выбрать из него пункт «Edit Data» (Редактировать данные), что приведет к появлению окна «Data Filters».



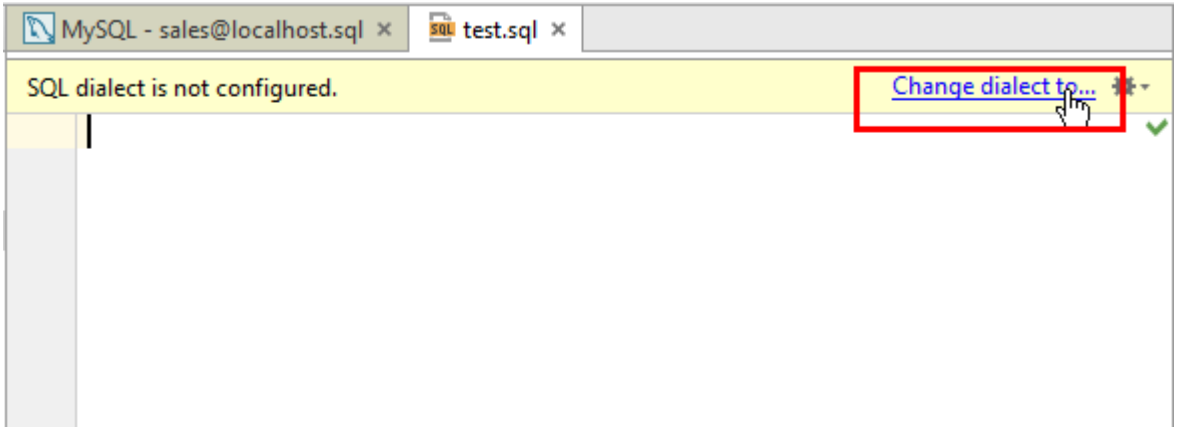
4. Щелкнем кнопку **«No Filter»** (Без фильтра), в результате чего в окне редактирования отобразится вкладка с данными таблицы, с которыми мы опять-таки можем совершенно спокойно работать практически также как и в клиенте СУБД.



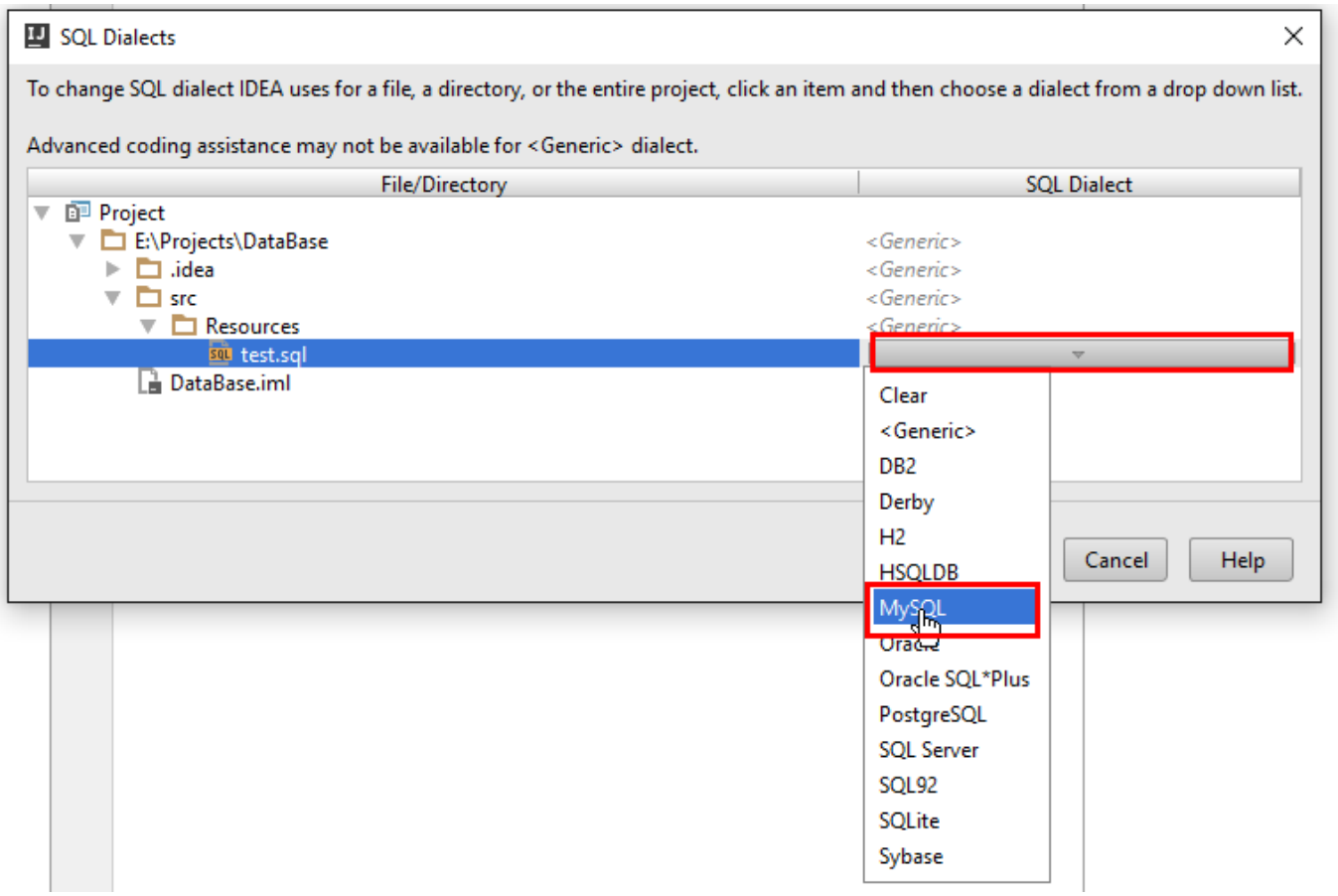
Подключение запросов

Естественным будет наше желание в дальнейшем выполнять в среде IIDEA SQL-запросы. Для создания запроса достаточно создать новый файл (**File ▸ New ▸ File**) с расширением sql.

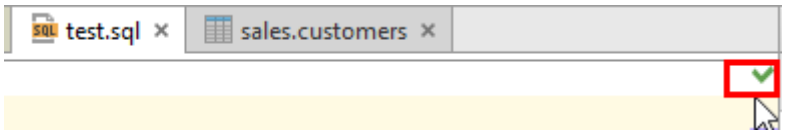
После создания первого такого файла необходимо будет определить для IIDEA диалект MySQL для дальнейшей работы.



Для этого строке с сообщением «SQL dialect is not configured» (SQL диалект не определен) щелкаем гиперссылку «Change dialect to…» (Изменить диалект на…), а затем, раскрыв большую кнопку списка, выбираем из него «MySQL».



После выполнения такого действия сообщение «SQL dialect is not configured» исчезает, а в правой части строки появляется символ, сигнализирующий о том, что все в порядке:



Получаем соединение с MySQL

Как было отмечено ранее, после создания проекта Maven при первом его открытии появится всплывающее окно с предупреждением о необходимости импортировать созданный проект.

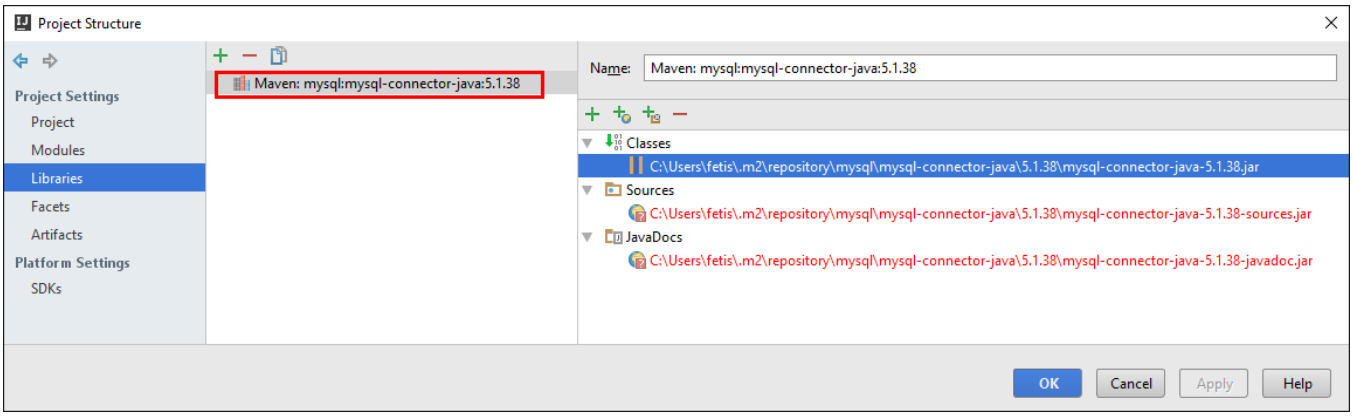
Выбор Enable Auto-Import необходим для того, чтобы библиотеки, подключаемые к проекту, автоматически скачивались со специальных репозиториев.

Для того чтобы работать с JDBC для MySQL, необходимо скачать jdbc mysql maven. Для этого следует перейти по ссылке http://mvnrepository.com/artifact/mysql/mysql-connector-java/5.1.6. В окне с перечнем версий выбрать, например, последнюю, щелкнув ее номер и скопировать содержимое <dependency>.

Во вкладке с уже существующим html-файлом добавляем парный тег <dependencies> и внутри него вставляем скопированное на сайте содержимое <dependency>. Это означает, что наш проект зависит от библиотеки из <dependency>, которое Maven должен после этого самостоятельно скачать.

Чтобы проверить, что библиотека уже подключена к проекту, необходимо:

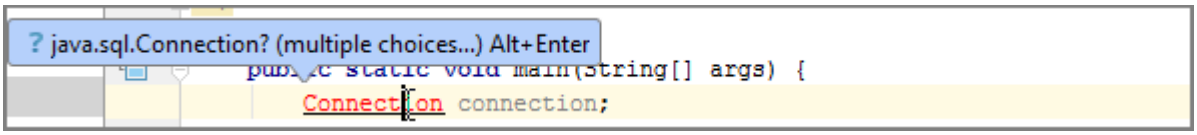
1. Выполнить команду **File ▸ Project Structure** что приведет к открытию окна «Project Structure».
2. В левой части окна выбрать пункт «Libraries» (Библиотеки).
3. Если в окне отображается пункт Maven: mysql:mysql-connector-java, то это и есть свидетельство того, что библиотека подключена Maven (текст «Maven» в начале строки как раз и говорит о том, что библиотека скачена именно Maven).



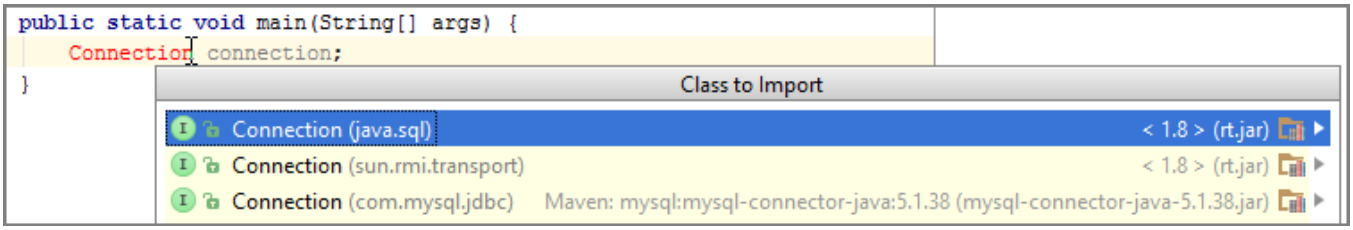
Для начала работы с базой данных приведем структуру проекта в соответствии с выбранным архетипом, для чего создадим в папке Java папку с именем, определенным при создании проекта в поле Groupid. В ней создается класс с именем, например, Main, а в нем метод public static void main(String[] args).

Дальнейшие действия происходят по такому алгоритму.

1. Необходимо определить переменную типа Connection.



При этом программа укажет на ошибку. Для ее устранения нажимаем предлагаемую программой комбинацию **<Alt>+<Enter>** и выбираем из первого списка пункт «Import class», а из следующего за ним – «Connection (java.sql)»:



Выполненные действия импортируют пакет

```
import java.sql.Connection;
```

и, соответственно, устраняют ошибку.

Вся дальнейшая работа с MySQL как раз и предполагает наличие этого пакета (библиотеки).

Переменная connection как раз и осуществляет соединение с базой данных, после чего становится возможным выполнение манипуляций с базою данных.

2. Указываем, какой jdbc-драйвер будет использоваться для взаимодействия с базой данных. Для этого создается класс Driver (также импортируя библиотеку) и создаем экземпляр класса:

```
Driver driver = new FabricMySQLDriver()
```

Заключить оператор необходимо в try/catch конструкцию (о чем свидетельствует указание на ошибку программы), поскольку возможно, что класс может отсутствовать, например, если библиотека не подключилась к проекту.

```
try {
        Driver driver = new FabricMySQLDriver();
} Catch (SQLException e) {
        e.printStackTrace();
}
```

3. Создается драйвер менеджер, с помощью которого регистрируется наш драйвер.

```
try {
        Driver driver = new FabricMySQLDriver();
        DriverManager.registerDriver(driver);
} Catch (SQLException e) {
        e.printStackTrace();
}
```

4. Для проверки корректности выполненных данных выполнить команду **Run ▸ Run 'Main'**. При отсутствии ошибок программа выведет сообщение

```
Process finished with exit code 0
```

После выполнения этих действий можно приступить к подсоединению к базе данных. Для этого воспользуемся методом getConnection из DriverManager. При этом возможны три комбинации аргументов для метода, например URL имя и пароль к базе данных (можно также URL и свойства). Для этих аргументов предварительно можно определить константы. Кроме того, по окончанию работы с соединением, оно должно быть закрыто:

```
private static final String URL = "jdbc:mysql://localhost:3306/sales";
private static final String USERNAME = "root";
private static final String PASSWORD = "1234";
...
try {
        Driver driver = new FabricMySQLDriver();
        DriverManager.registerDriver(driver);
        connection = DriverManager.getConnection(URL, USERNAME, PASSWORD);
        connection.close();
```

Установленное соединение позволяет в дальнейшем выполнять запросы к базе данных. Одновременно исчезает необходимость загрузки сервера MySQL.

Выполнение статических запросов

Статические запросы – это запросы, в которых отсутствуют переменные.

При помощи метода ***execute()*** класса ***Statement*** можно выполнять операции вставки новой записи, получения данных.

```
statement.execute("INSERT INTO customers VALUES (2009, 'Bob', 'Nizhin', 100, 1002);");
```

Естественно, после добавления в код строки выполнить команду **Run**. Понятно, что после выполнения программы в окне консольного вывода должно появится сообщение:

```
Process finished with exit code 0
```

При этом изменения отобразятся во вкладке с таблицей только после обновления данных в таблице, выполнение которого осуществляется щелчком кнопки  («Reload page»). Все вышесказанное относится и ко всем дальнейшим действиям, связанным с модификацией данных в таблицах.

При помощи метода ***executeUpdate()*** класса ***Statement*** выполняются все операции модификации данных:

```
statement.executeUpdate("UPDATE customers SET City='Sun Jose' WHERE Snum = 1007");
statement.executeUpdate("DELETE FROM customers WHERE City ='Nizhin';");
```

Метод возвращает число записей, в которых были произведены изменения SQL-оператором.

При пакетной обработке можно выполнить за один раз несколько SQL-операторов. Для этого используются методы ***addBatch()***, аргументом которого является SQL-оператор и ***executeBatch()*** (без аргументов).

```
statement.addBatch("INSERT INTO customers VALUES (2009, 'Bob', 'Nizhin', 100, 1002);");
statement.addBatch("INSERT INTO customers VALUES (2009, 'Bob', 'Nizhin', 100, 1002);");
statement.executeBatch();
```

При помощи метода isClosed() проверяется закрытие statement.

```
boolean statamentClose = statement.isClosed();
```

Вывод результатов запросов

https://www.youtube.com/watch?v=3AqWyO86f_U

Для выполнения запросов при помощи оператора SELECT используется метод ***executeQuery()***.

```
statement.executeQuery("SELECT * FROM customers");
```

Для вывода результатов выполнения запроса необходима модель ответа запроса, называемая ResultSet:

```
ResultSet resultset = statement.executeQuery(query);
```

При построении запроса используется цикл для обхода данных по всем строкам таблицы. Для получения данных используются методы, начинающиеся с текста ***get***, которые должны совпадать с типом поля. Например, для поля с целочисленным типом данных используется метод ***getInt()***, а для данных типа строка – ***getString()***.

Аргументом метода ***get*** может быть как имя поля, так и его порядковый номер в SQL-запросе (в операторе SELECT). Предпочтительней использовать имя поля, поскольку изменение порядка и (или) состава полей в SELECT никак не отразится на программном коде.

```
String query = "SELECT Cnum, Cname, Rating FROM customers";
…
 while (resultset.next()){
 String Cnum = resultset.getString(1);
 String Cname = resultset.getString("Cname");
 int Rating = resultset.getInt(3);
```

```
 System.out.println(Cnum+" "+ Cname+" "+Rating);
}
```

В SQL-запросах можно использовать любые корректные конструкции.

Вывод можно также организовать и путем переопределения метода toString класса Object. Не получилось.

Выполнение динамических скомпилированных запросов

https://www.youtube.com/watch?v=o5OdDWfSAWQ

```
private static final String INSERT_NEW = "INSERT INTO salespeople VALUES (?,?,?,?)";
...
public static void main(String[] args)
 {
 Connection connection = null;
 PreparedStatement preparedStatement = null;
 try {
 Driver driver = new FabricMySQLDriver();
 DriverManager.registerDriver(driver);
 connection = DriverManager.getConnection(URL, USERNAME, PASSWORD);

 preparedStatement = connection.prepareStatement(INSERT_NEW);
```

Значения в SQL-операторе в строке String представлены в виде «?». Это означает, что на их место необходимо подставить какие-то конкретные значения. Их можно представить как переменные в запросе. Для подстановки значений существует ряд методов, в основном в preparedStatement. Они начинаются с текста set, после чего следует тип данных в поле. Первым параметром таких методов является очередность (номер позиции в SQL-запросе) или название поля.

```
preparedStatement.setString(1, "1010");
...
preparedStatement.setBigDecimal(4, BigDecimal.valueOf(0.2d));
preparedStatement.execute();
```

Выполняется SQL-оператор методом ***execute()***.

Опять-таки для отображения произведенных изменений следует на вкладке с таблицей, в которую вносились изменения, щелкнуть кнопку 

«Reload» (Обновить).